# Counting-by-Detection with Three-dimensional Fully Convolutional Networks



Présenté le 25 Janvier 2018.
À la Faculté des Computer and Communication Sciences IC.
Laboratoire Computer Vision.
École Polytechnique Fédérale de Lausanne.
Pour l'obtention la note dans un projet de master semestre.

Presented at 25 Genuary 2018.
At Escola Tècnica Superior d'Enginyeria de Telecomunicacions de Barcelona.
To obtain the grade of the final project.

Advisors:
Pablo Marques Neila, coordinateur de projet à l'EPFL.
*pablo.marquezneila@epfl.ch*
José Adrian Fonellosa, advisor in UPC.
*jose.fonollosa@upc.edu*

**Sergi Mas Pujol**
**sergimasp@gmail.com**
**Lausanne, EPFL, 2018**

One day, we'll be old and think about the stories
that we could have lived.

To my parents...

A la meva mare per continuar lluitant els derrers anys per fer-ho possible i mantenir-se sempre al meu costat.

Al meu pare per donar-ho tot per poder arribar a aquest punt i aquest moment.

To my advisor...

A Pablo Marquez Neila, por lo oportinudad de realizar este proyecto, por el conocimiento transmitido, la ayuda proporcionada y el soporte dado durante la realización del mismo.

# Abstract

The project goal is to count and detect instances in an image using deep learning. We target the regime where object detectors need to work reliably in scenarios with crowding, overlapping or instances with a small size.
To estimate the objects, two approaches have been used: counting-by-detections and counting-by-density-estimation.
We concluded that objects could be rated simply, accurately and efficiently using a counting system and a classification model. To support this statement, we compare different experiments using different FCNs.
Finally, we propose a quantitative and qualitative method, to evaluate the network. The results are demonstrated on a variety of visual material, including graphs, microscopy and density images.

Keywords: The acronym FCN means Fully Connected Network.

# Resum

L'objectiu del projecte és explicar i detectar instàncies en una imatge mitjançant l'aprenentatge profund. El nostre objectiu és un entorn en el qual els detectors d'objectes han de treballar de manera correcta en escenaris abarrotats, amb superposicions o instàncies amb una mida petita.
Per estimar els objectes, s'han utilitzat dos enfocaments: recompte per detecció i recompte per estimació de densitat.
Concloem que els objectes poden classificar-se de manera simple, precisa i eficient utilitzant un sistema de detecció i un model de classificació. Per donar suport a aquesta afirmació, comparem diferents experiments utilitzant diferents FCN.
Finalment, proposem un mètode quantitatiu i qualitatiu per avaluar la xarxa. Els resultats es demostren en una varietat de material visual, incloent gràfics, microscòpia i imatges de densitat.

Paraules clau: L'acrònim FCN significa Xarxa Totalment Connectada

# Resumen

El objetivo del proyecto es contar y detectar instancias en una imagen mediante el aprendizaje profundo. Nuestro objetivo es un entorno en el que los detectores de objetos deben trabajar de manera correcta en escenarios abarrotados, con superposiciones o instancias con un tamaño pequeño.
Para estimar los objetos, se han utilizado dos enfoques: recuento por detección y recuento por estimación de densidad.
Concluimos que los objetos pueden clasificarse de manera simple, precisa y eficiente utilizando un sistema de conteo y un modelo de clasificación. Para apoyar esta afirmación, comparamos diferentes experimentos usando diferentes FCN.
Finalmente, proponemos un método cuantitativo y cualitativo para evaluar la red. Los resultados se demuestran en una variedad de material visual, incluidos gráficos, microscopía e imágenes de densidad.

Palabras clave: El acrónimo FCN significa Red Completamente Conectada.

# List of Figures

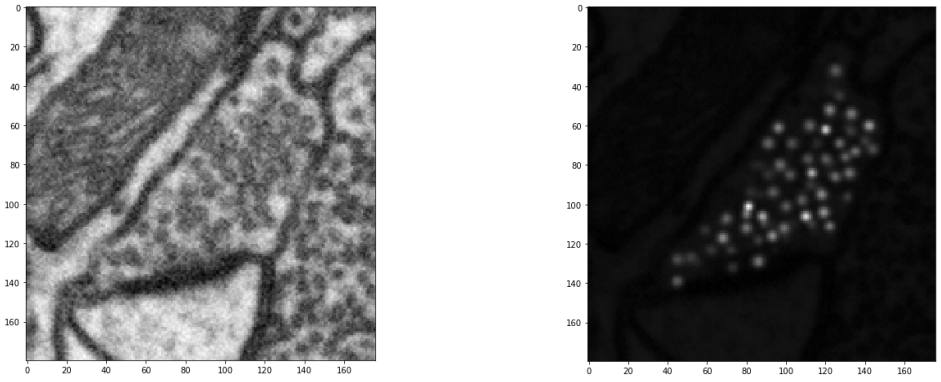
# List of Tables

# Contents

# 1 Introduction

Counting objects in images is a common problem with many real-world applications: counting people in crowds for automatic video surveillance systems, or counting cells in microscopy imaging of biological tissue are two representative examples.

With the advent of SBFSEM microscopes in 2004, it became possible to generate high-resolution 3D images of small samples of brain tissue. This lead to an important boost for neuroscience: neuroscientists were able for the first time to visualise small-scale brain structures, such as mitochondria, synaptic joints or synaptic vesicles, in 3D. These images potentially contain valuable information for the neuroscientists: the size of mitochondria, area of synaptic junctions, shape of dendritic spines and axonal boutons, or number and distribution of synaptic vesicles.




(a) (b)

Figure 1.1 – **Synaptic vesicles.** High-resolution 3D images obtained using SBFSEM microscopes. (a) Brain structures from the striatum. (b) The same image emphasizing the synaptic vesicles.

Synaptic vesicles have substantial biological importance. They are the elements of the brain that permit transmission of information. Vesicles are small elements of ~40nm diameter found in neuronal axons. They contain the neurotransmitters that travel through the synaptic joints from the axons to the dendrites of another neuron. Counting the synaptic vesicles and knowing their spatial distribution are essential pieces of information for biologists.

However, given a large amount of image data generated by EM microscopes and the huge amount of synaptic vesicles, it is unfeasible for neuroscientists to annotate them one by one of these 3D images manually. Instead, automatic image processing seems to be necessary for this task.

In this work, we aim to develop a 3D object counting method for counting synaptic vesicles in electron-microscopy images of brain tissue. Although, the system developed could be used in other scenarios with similar characteristics.

Exist two classic proceedings to count objects in images: counting-by-detection and counting-by-density-estimation. The first approximation uses the coordinates of the synaptic vesicles centres, and the second, as the name suggests, use the density estimation.

In this work, we use a Fully Convolutional Network (FCN) [1] known as the U-Net [2] depicted by Fig. 3.3 , for counting-by-detection and counting-by-density-estimation.

Ultimately, our contributions can be summarised as follows:

- Define a 3D counting method for counting-by-detection using a FCN.
- Present a new process to estimate the precision and recall of counting-by-detection methods.
- Evidence the performance of the method experimentally.

## 1.1 Work plan with tasks, milestones and a Gantt diagram.

The project is divided into different parts. The stages were assigned in this order to increase the learning capacity of the models.
The main stages are:

1. Study machine learning and neural networks.
2. Improve skills on internal tools.
3. Prepare the training data.
4. Develop Neural Networks that works with 2D.
5. Develop Neural Networks that works with 3D.
6. Documentation and Conclusions.

If you want more information about the work plan, the task description, the milestones or the gantt diagram, refer to the appendix.
Or you can find an extended explanation in the Project Critical Review (PCR) that was delivered to the advisors of the project.

The main issues appear in stage 5. We decided to didn't go farther with the network that works with density.

# 2 Related Work

As detailed in the introduction, the existing 3D counting classics methods for images can be roughly categorised into counting-by-detection and counting-by-density-estimation.

**Counting-by-detection:** This assumes the use of a visual object detector that localises individual object instances in the image. Most current object detectors operate in two stages: first producing a real-valued confidence map; and second, given such a map, a further thresholding and non-maxima suppression steps are needed to locate peaks corresponding to individual instances. Other generative approaches avoid non-maxima suppression by reasoning about relations between object parts and instances.

**Counting-by-density-estimation:** The paper [3] proposes a novel specific distance metric MESA between density functions used as a loss in their framework. They claim that this distance possesses two highly desirable properties: robustness and computability.
Arteta et al. [4] introduce a system for counting multiple instances of an object class in crowded scenes using ridge regression. They present the density estimation results in an intuitive way to know where further annotations may be required.
Both [3] [4] present a solution that speeds up the learning of object densities.

The two works claim that counting-by-density-estimation is a better solution than counting-by-detection and that counting-by-detection approaches are geared towards situations with a small number of objects in images, require time-consuming inference and assume that objects tend to be uniform and disconnected from each other. They conclude that methods in these groups deliver accurate counts when their underlying assumptions are met but are not applicable in more challenging situations.
In this work we demonstrate that this conclusion is not true using deep learning, because our method is able to predict with a high level of accuracy the location of vesicles even when they are touching or overlapping each other in the images.

# 3 Approach

## 3.1 Pre-processing of the available training data.

**Counting-by-detection**

We use a fully convolutional network (FCN) [1] as our trainable model, as it is currently among the best and most widely used architectures for segmentation in both natural and biomedical images. FCN map images to images, and therefore can be trained in an end-to-end setting.

In this project, we use a FCN known as the U-Net [2] to implement our 3D counting method. Instead of training the network for segmentation of whole vesicles, we use it to segment only their centres. This is a simple way to perform object detection simultaneously over the pixels of the whole image.

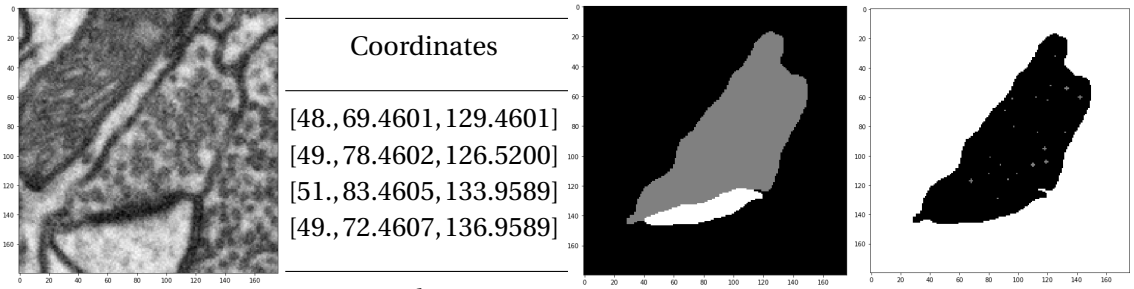



Figure 3.1 – **Training labels.** Graphical example of the process to obtain the training labels used in the FCN. (a) is the EM images, (b is the coordinates of the centres of the vesicles, (c) is the mask of the axonal bouton and (d) is the label volume.

To train the network to perform object centre detection, we need to pre-process the available training data. Our raw training data consists of several stacks of EM images depicting axonal boutons containing vesicles (Fig. 3.1(a)). Each of these boutons is manually annotated by a biologist. The manual annotations contain the coordinates of the centres of the vesicles

(Fig.3.1(b)) and a mask indicating the location of the axonal bouton in the image ( 3.1(c)). With this information we build a label volume (Fig. 3.1(d)) used to train the U-Net following the procedure depicted in Fig. 3.1. The voxels corresponding to the coordinates of the centres and a few voxels in their neighbourhoods are set to the label "centre". Voxels surrounding these *central voxels* are set to the label "ignore", along with the voxels outside the axonal bouton given by the annotated mask. The rest of the voxels are assigned the label "background".

**Data:** $original\_image, vesicles\_coordinates, mask\_axon$
original_centres_img = centres_activation(original_image.shape, vesicles_coordinates)
dilated_centers_img = 3D_dilation(original_centres_img)
    compute_euclidean_dinatance()
    set_threshold()
    return dilated_centers_img
label_volume = uncentanty_zone(mask_axon)
**Result:** $label\_volume$

**Algorithm 1:** 3D-dilation of the vesicles centres.

Once we have generated all the label images, we use them to train the U-Net.

At test time, we apply non-maxima suppression over the output of the network to obtain the coordinates of the predicted centres. Algorithm 2 gives the details of this procedure.

**Data:** $unet\_prediction$
dilated_prediction = grey_dilation(unet_prediction)
centres_prediction = logical_and(unet_prediction, dilated_prediction)
**Result:** $centres\_prediction$

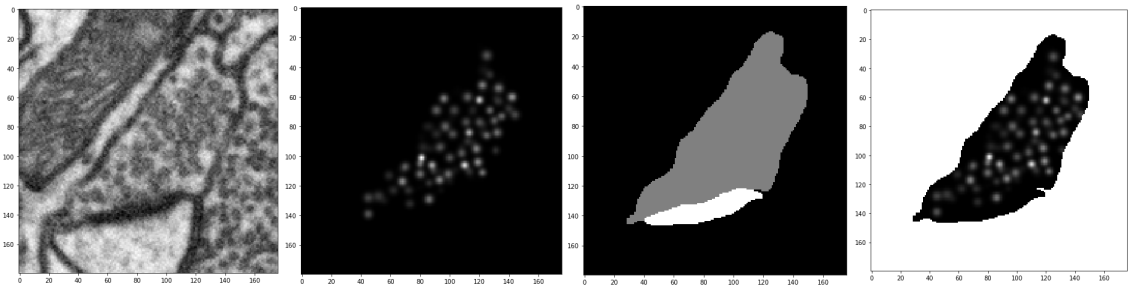
**Algorithm 2:** Non-maxima suppression implementation.

### Counting-by-density-estimation

The U-Net used can understand a scenario where the main data is associated with the density distribution of the synaptic vesicles in the axonal bouton. In our raw data, we have the density distribution and we replicate the conditions exposed before to compare objectively the results obtained.

In this approach, our raw data consist of several stacks of EM images depicting axonal boutons containing vesicles (Fig. 3.2(a)). The manual annotation contains the density of the synaptic vesicles (Fig 3.2(b)) and masks indicating the location of the axonal bouton (Fig 3.2(c)). With this information we build a labelled volume (Fig 3.2(d)) used to train the U-Net following the procedure depicted in 3.2. The voxels corresponding to the vesicles are set to the label "vesicle". Voxels surrounding these *vesicles* are set to the label "ignore", along with the voxels

outside the axonal bouton given by the annotated mask. The rest of the voxels are set to the label “background”.



(a) (b) (c) (d)

Figure 3.2 – **Training labels density.** Graphical example of the process to obtain the training labels used in the FCN. (a) Is the EM images, (b) is the density of synaptic vesicles, (c) is the mask of the axonal bouton and (d) is the label volume.

## 3.2 U-Net Architecture

U-Net[2] is a deep convolutional neural network[5] for fast and precise segmentation of images and is considered a sliding-window convolutional network. As shown in Fig. 3.3, it is a fully-convolutional architecture that includes links between non-consecutive layers. It performs a series of convolutions, followed by a nonlinear activation function (ReLU), and pooling operations to reduce its spatial resolution, where we gradually increase the ’what’ and at the same time decrease the ’where’. Followed by up convolutions to produce an output of the same resolution as the input image. At the output, each channel encodes the probability of a specific joint to be observed at a given image location.

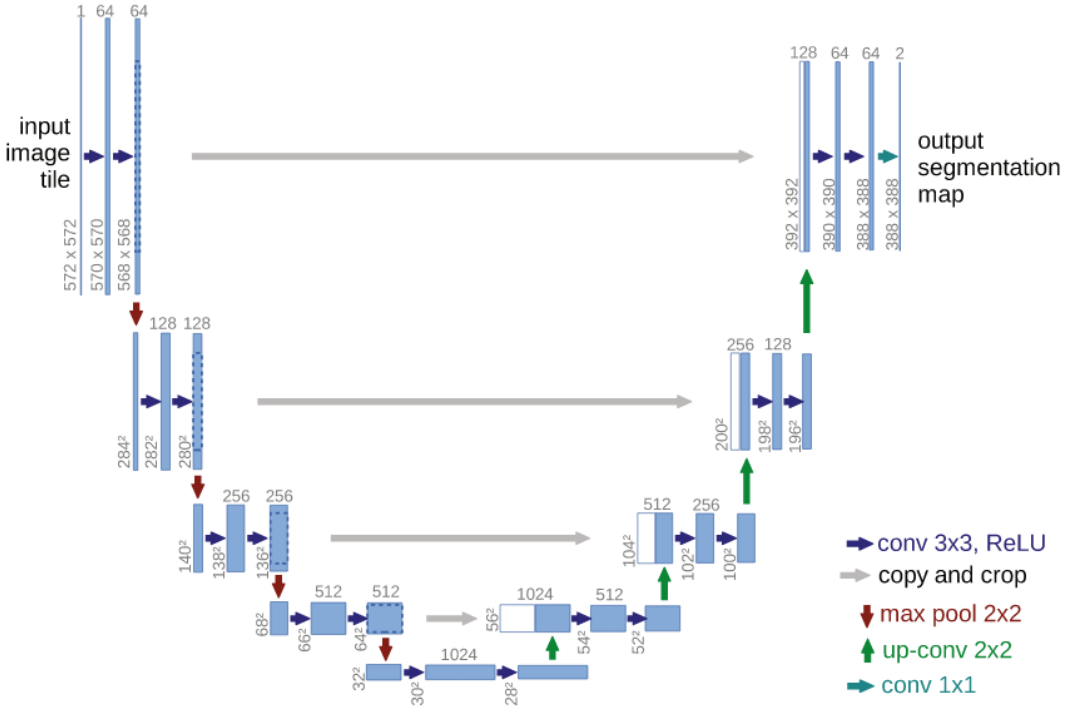



Figure 3.3 – **U-Net architecture.** Example for 572 × 572 pixels of the input image where each blue box corresponds to a multi-channel feature map. The arrows denote the different operations, the number of channels is denoted on top of the box, the xy-size is provided at the lower left edge of the box and the white boxes represent copied feature maps

## 3.3 Loss function

**Cross-entropy**

We train the U-Net minimizing the binary cross-entropy loss [6]. For a single pixel, the cross-entropy is defined as

$$L(y^{(i)}, \hat{y}^{(i)}) = -(1 - y^{(i)}) \cdot \log(1 - y^{(i)}) - y^{(i)} \cdot \log y^{(i)}, \tag{3.1}$$

Where $y^{(i)}$ is the ground-truth label of pixel $i$ and $\hat{y}^{(i)}$ is the predicted probability for that pixel given by the network. For the training, the loss is averaged over all pixels and training samples. The neural network is therefore trained as a pixel-wise classifier. By minimizing the loss function, we are encouraging our model to assign higher probability values to the right labels across training examples.

**Density pixel**

In the second approach of the project, we try to develop another FCN that works with densities.

This new scenario poses two great challenges: Is necessary to use a network that works with regression model and develop a new 'loss function'.

$$L(y^{(i)}, \hat{y}^{(i)}) = \sum (w * \hat{y}^{(i)} - y^{(i)})^2 \tag{3.2}$$

Where $w$ is the weights of the network, $\hat{y}^{(i)}$ is the predicted probability given by the network and $y^{(i)}$ is the ground-truth label.

The neural network is therefore trained as a regression predictive modelling. Taking any decision using the differences between any pixel.

**Density map**

In all of this project, we are using "EMSA", explained in the section 3.4.2 paragraph 2, to generate random boxes to calculate some parameters. The last improvement is to try to use these boxes to compute the 'loss function'.

$$L(y^{(i)}, \hat{y}^{(i)}) = \sum (w * \hat{y}^{(i)}[r1:r2, c1:c2] - y^{(i)}[r1:r2, c1:c2])^2 \tag{3.3}$$

Where $w$ is the weights of the network, $\hat{y}^{(i)}$ is the predicted probability given by the network, $y^{(i)}$ is the ground-truth label and the [r1:r2, c1:c2] are the limits of the boxes created using "emsa".

The neural network is trained as a regression model. For the training, the loss is averaged over the pixels inside of the random boxes.

## 3.4 Evaluation metrics

To report results of our experiments, we use two sets of metrics. The first is restricted to counting-by-detection methods while the second can be used in both counting-by-detection and counting-by-density-estimation, and therefore it is more suitable for comparison purposes.

### 3.4.1 Detection metrics

The detection metrics consist of the standard metrics *precision* (P), *recall* (R) and *F1-Score* (F1).

$$P = \frac{TP}{TP+FP} \qquad R = \frac{TP}{TP+FN} \tag{3.4}$$

$$F1 = 2\frac{1}{1/P+1/R} = 2\frac{P*R}{P+R} \tag{3.5}$$

These metrics require cataloguing each element as true positive (TP), false positive (FP) and false negative (FN) in the prediction and in the ground-truth.

In counting-by-detection, we have two sets of coordinates at test time. The coordinates of the centres of the vesicles given in the ground-truth and the coordinates predicted by the U-Net after non-maxima suppression. It is not trivial to determine which elements are TP, FP and FN from these two sets. To solve this problem, in Algorithm 3 we define a procedure to find correspondences between two sets of coordinates. The basic idea is finding the correspondences that minimize the sum of distances between matching coordinates. This problem is equivalent to finding a minimum weight perfect matching in bipartite graphs:

$$min_X \sum_i \sum_j C_{ij} X_{ij}, \tag{3.6}$$

Where $C_{ij}$ is the cost of matching elements $i$ and $j$ (in our case that is the distance between coordinate $i$ from ground-truth and coordinate $j$ from prediction) and $X_{ij}$ is a boolean matrix where $X_{ij}$ iff element i is assigned to element j. This can be solved efficiently with the Hungarian method [7].

After a set of correspondences is found, we remove correspondences between pairs of points that are farther than a threshold distance (4 voxels in our case).

Once we found the correspondences, the resulting set is the TP, the detections without correspondences are the FP and the elements of the ground-truth without correspondences are the FN. With this, we compute precision, recall and the F1-Score.

**Data:** $testing\_coords, predicted\_coords, match\_distance$
w = distance_matrix(testing_coords,predicted_coords)
pairs = optimize.linear_sum_assignment(w)
**for** pair **in** pairs:
    **if** distance(pair) <= match_distance
        correspondences.append(pair)
precision = **len**(correspondences)/predicted_coords
recall = **len**(correspondences)/testing_coords
tp = **len**(correspondences)
tp_fp = **len**(predicted_coords)
tp_fn = **len**(testing_coords)
**Result:** $precision, recall, tp, tp\_fp, tp\_fn$

**Algorithm 3:** Correspondences between coordinates.

Where in Algorithm 3: $tp$ is the true positive, $tp\_fp$ is the sum of true positives and false positive and $tp\_fn$ is the sum of true positives and false negative.

### 3.4.2 Density estimation metrics

This set of metrics are based on density estimation methods, although it can also be used in counting method.

To compute this metric, we use "EMSA" which permit us to generate random boxes in the ground-truth and in the prediction to sum the content. The mean of the difference divided by the volume provides us an estimation of the prediction error for voxel.

$$average\_density\_error = \frac{1}{n}\sum_{i=0}^{n}\frac{true\_sums(i) - pred\_sums(i)}{volume(i)}[vesicles/voxel] \quad (3.7)$$

Where in 3.7, "true_sums" and "pred_sums" correspond with the number of vesicles in the original set and in the prediction. The "volume" corresponds to the volume of each random box generated.

# 4 Experiments

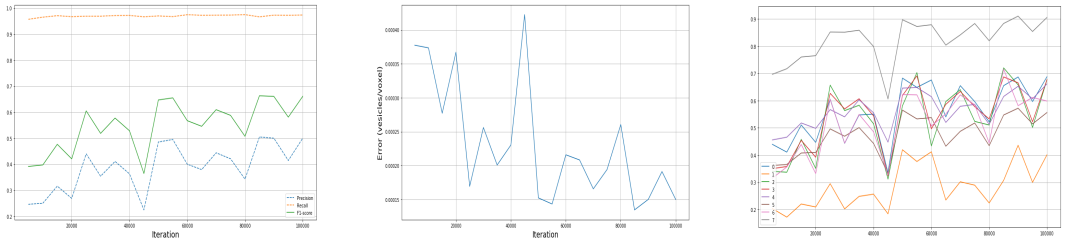
**Counting-by-detection**

For our experiments we used a dataset with 15 stacks of EM images from different parts of a rat brain. We split the dataset in 7 stacks for training and 8 for testing. This split, the size of the stacks and the number of synaptic vesicles contained in each of them are listed in Table 4.1.

| Data set name | Size | Number of vesicles |
|---|---|---|
| hipp_testing/Bouton 2 | (165, 272, 177) | 417 |
| hipp_testing/Bouton 3 | (141, 308, 657) | 284 |
| glutamate/Bouton 1 | (162, 333, 184) | 951 |
| glutamate/Bouton 2 | (142, 413, 388) | 920 |
| striatum_training/Bouton 1 | (156, 181, 154) | 640 |
| striatum_training/Bouton 2 | (85, 283, 175) | 235 |
| striatum_training/Bouton 3 | (126, 200, 281) | 429 |
| hipp_training/Bouton 1 | (165, 141, 266) | 176 |
| hipp_testing/Bouton 1 | (165, 309, 288) | 196 |
| glutamate/Bouton 3 | (207, 138, 450) | 468 |
| glutamate/Bouton 4 | (165, 231, 331) | 587 |
| striatum_testing/Bouton 1 | (117, 450, 165) | 415 |
| striatum_testing/Bouton 2 | (177, 450, o225) | 560 |
| striatum_testing/Bouton 3 | (208, 205, 255) | 789 |
| striatum_testing/Bouton 4 | (318, 180, 176) | 1053 |

Table 4.1 – **Data set description.** Name, size (number of slices, height, width) and number of vesicles in each stack of the dataset. At the top the training datasets (7 stacks) and at the bottom the testing datasets (8 stacks).

As a baseline and for comparison purposes, we first trained a 2D U-Net to perform independent slice predictions. The slice predictions were combined in the non-maxima suppression step, which is always performed in 3D. We call this experiment **UNet2D-no-refinement**. The results are listed in the first row of Table 4.2, Table 4.3 and in Figure 4.1. While the precision score is reasonably good, the recall scores obtained in this experiment are poor.



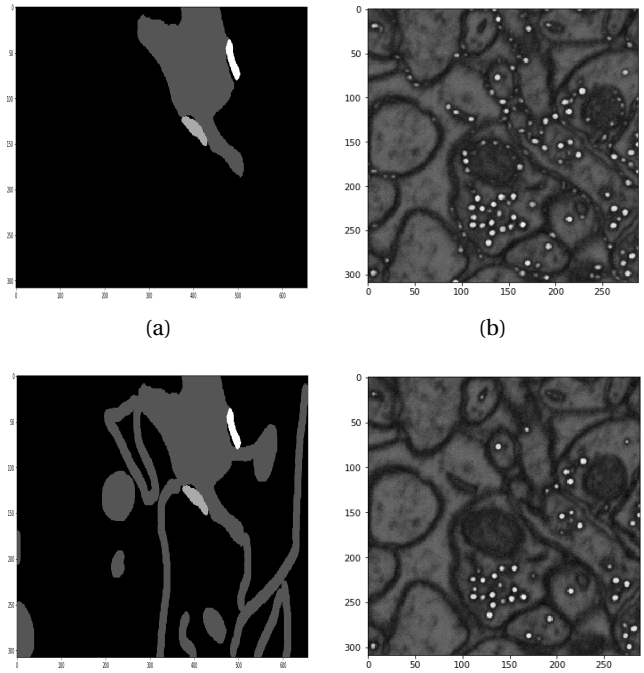

(a) (b) (c)

Figure 4.1 – **Detection 2D-no-refinement.** Best results obtained using images in 2D and counting by detection. (a) is the recall/f1-score/precision,(b) the error and (c) the specific f1-score in all testing datasets.

A deeper analysis showed a large amount of false positive predictions in areas that hardly resemble a vesicle, as seen in Figure 4.2(b). This suggested a lack of background labeled data. As shown in Figure 4.2(a) the initial annotated data did not include enough membranes or mitochondria to make the network prediction negative in these areas. To solve this problem, we manually extended the background label in several slices of the training data. Figure 4.2(c) shows one of these slices with extended background annotations.

We repeated the same training with the new extended annotations in the experiment **UNet2D**. The results are listed in Table 4.2, Table 4.3 and in Figure 4.3. Fig. 4.2(d) shows the predictions on the same slice than in Fig. 4.2(b), but obtained with the new training data( Fig. 4.2(c)). As can be seen, the performance increased remarkably with the new annotations.

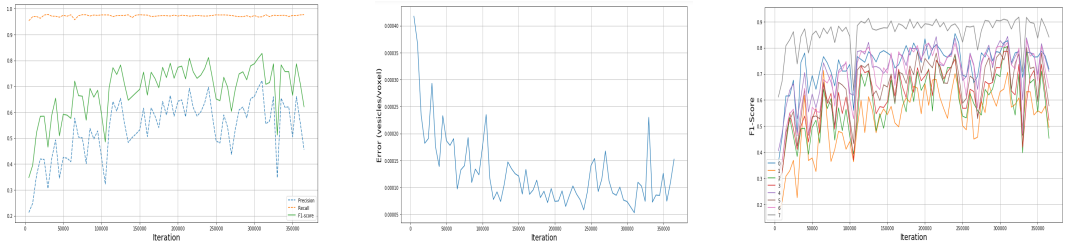
With the new extended data we also trained a 3D U-Net in the experiment **UNet3D**. Results are shown in Table 4.2, in Table 4.3 and in Figure 4.4. Again, using 3D information notably improved the performance of our predictions, increasing the F1-Score (Fig. 4.5) in 9 percent points.



(a) (b)

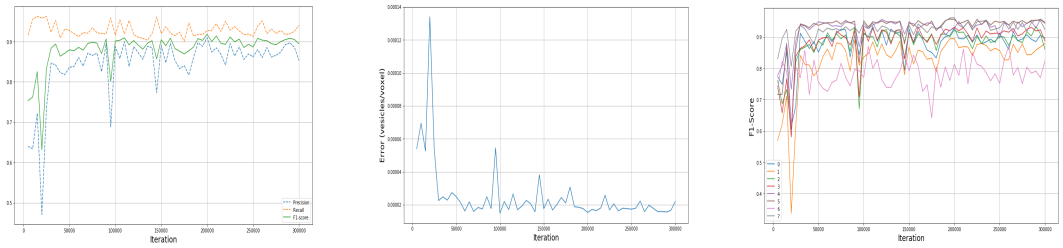
(c) (d)

Figure 4.2 – **Mask Expansion.** Permits to reduce the number of FP. (a) Original mask in training, (b) initial prediction, (c) mask expansion in training and (d) Final prediction.




(a) (b) (c)

Figure 4.3 – **Detection 2D.** Best results obtained using images in 2D and counting by detection. (a) is the recall/f1-score/precision,(b) the error and (c) the specific f1-score in all testing datasets.




(a) (b) (c)

Figure 4.4 – **Detection 3D.** Best results obtained using images in 2D and detection. (a) is the recall/f1-score/precision, (b) the error and (c) the specific f1-score in all testing datasets.

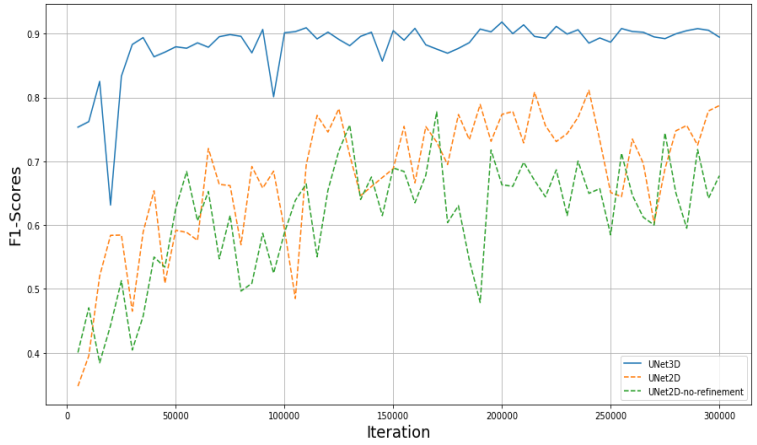
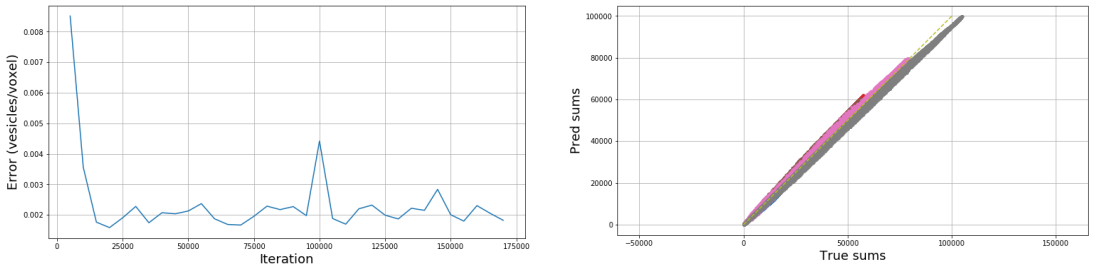



Figure 4.5 – **F1-Score.** Graphical comparative between the f1-Score obtained in the different experiments in counting-by-detection.

### Counting-by-density-estimation

Once we have the results using counting-by-detection we train a density U-Net to compare and observe the results. In this approach the baseline is a U-Net with a classic loss function (Eq: 3.2). We call the experiment **UNet-density-pixel**. The results are listed in Table 4.2, Table 4.3 and in the Fig. 4.6

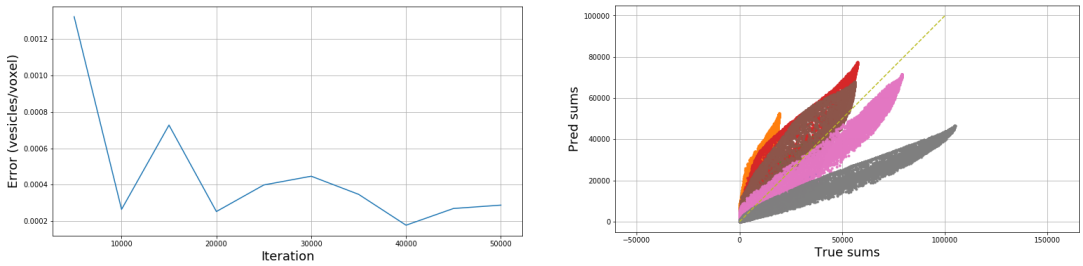



(a) (b)

Figure 4.6 – **Density Pixel.** Best results obtained using counting by density estimation and the loss function 'density pixel'. (a) Is the error and (b) the true_sums/pred_sums.

Where in Fig. 4.6(b), "true_sums" and "pred_sums" correspond with the number of vesicles in the original set and in the prediction. Each color represent a different testing dataset.

With the previous results we try another experiment called **UNet-density-map**. In order to improve the results we use the lost function 3.3. The results are listed in Table 4.2 and in the Fig. 4.7




(a) (b)

Figure 4.7 – **Density Map.** Best results obtained using counting by density estimation and using the loss function "density map". (a) Is the error and (b) the true_sums/pred_sums.

Where in Fig. 4.7(b), "true_sums" and "pred_sums" correspond again with the number of vesicles in the original set and the prediction. Each colour represents a different testing dataset.

Unfortunately, the new loss function (Eq: 3.3) proposed does not improve the training and the results.

## 4.1 Summary table of results

In the next table, we expose the best quantitative results obtained in the different experiments and then the results in each testing stack.

| Experiment | Precision | Recall | F1-Score | Error [vesicles/voxel] |
|---|---|---|---|---|
| UNet2D-no-refinement | 74.50% | 95.72% | 83.79% | 5.0771e-05 |
| UNet2D | 82.16% | 97.78% | 82.70% | 5.291e-05 |
| UNet3D | 90.98% | 96.23% | **91.80%** | **1.488e-05** |
| UNet-desnity-pixel | - | - | - | **5.9851e-06** |
| UNet-density-map | - | - | - | 1.0835e-3 |

Table 4.2 – **Results.** Best quantitative results obtained in the deferents experiments.

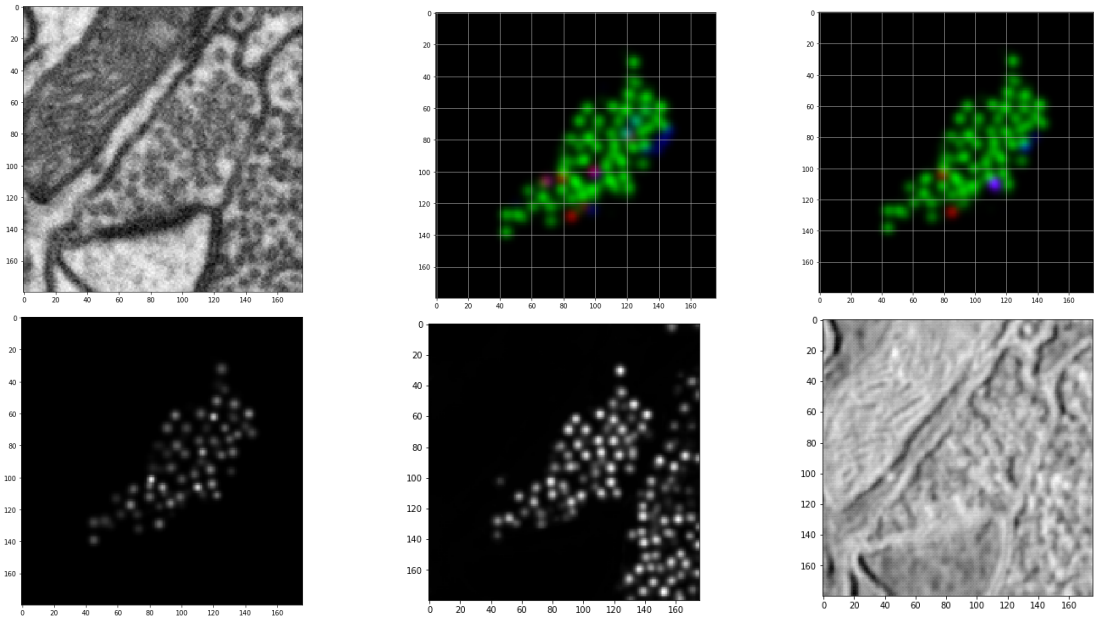
| Experiment | Metrics | hipp_train/1 | hipp_test/1 | glutamate/3 | glutamate/4 | striatum_test/1 | striatum_test/2 | striatum_test/3 | striatum_test/4 |
|---|---|---|---|---|---|---|---|---|---|
| UNet2D-no-refinement | Precision | 0.4956 | 0.2325 | 0.4159 | 0.4197 | 0.4288 | 0.3149 | 0.4366 | 0.7450 |
| | Recall | 0.9659 | 0.9540 | 0.9829 | 0.9982 | 0.9879 | 0.9910 | 0.9695 | 0.9572 |
| | F1-Score | 0.6551 | 0.3740 | 0.5844 | 0.5910 | 0.5981 | 0.4780 | 0.6021 | 0.8379 |
| | Error | 9.5599e-05 | 1.7242e-04 | 1.6506e-04 | 1.6654e-04 | 2.0739e-04 | 2.8990e-04 | 2.6270e-04 | 5.0771e-05 |
| UNet2D | Precision | 0.6164 | 0.421 | 0.5654 | 0.5588 | 0.6960 | 0.5902 | 0.7091 | 0.832 |
| | Recall | 0.9772 | 0.9285 | 0.9871 | 0.9948 | 0.9879 | 0.9928 | 0.9670 | 0.9563 |
| | F1-Score | 0.7560 | 0.5796 | 0.7190 | 0.7156 | 0.8167 | 0.7403 | **0.8182** | 0.8899 |
| | Error | 6.681e-05 | **7.001e-05** | 9.111e-05 | 9.235e-05 | **6.176e-05** | 9.513e-05 | 8.335e-05 | 3.705e-05 |
| UNet3D | Precision | 0.8301 | 0.8373 | 0.8238 | 0.8457 | 0.9207 | 0.9212 | 0.7904 | 0.9290 |
| | Recall | 0.9431 | 0.8928 | 0.9594 | 0.9897 | 0.9518 | 0.9821 | 0.7743 | 0.9221 |
| | F1-Score | **0.8829** | **0.8641** | **0.8864** | **0.9120** | **0.9360** | **0.9507** | 0.7823 | **0.9256** |
| | Error | **1.9179e-05** | 7.4524e-06 | **2.6824e-05** | **2.5556e-05** | 9.5656e-06 | **1.2061e-05** | **2.2098e-05** | **1.8420e-05** |
| UNet-desity-pixel | Error | **1.6503-05** | **5.9853e-06** | **7.712e-06** | **6.922e-06** | **1.221e-05** | **8.200e-06** | **3.230e-05** | **6.270e-05** |
| UNet-density-map | Error | 3.197e-03 | 3.616e-03 | 2.3496e-03 | 3.5362e-03 | 2.0055e-03 | 2.9533e-03 | 2.7878e-03 | 1.0835e-03 |

Table 4.3 – **Results in all stacks.** Results obtained during testing in all the stacks.

## 4.2 Qualitative results

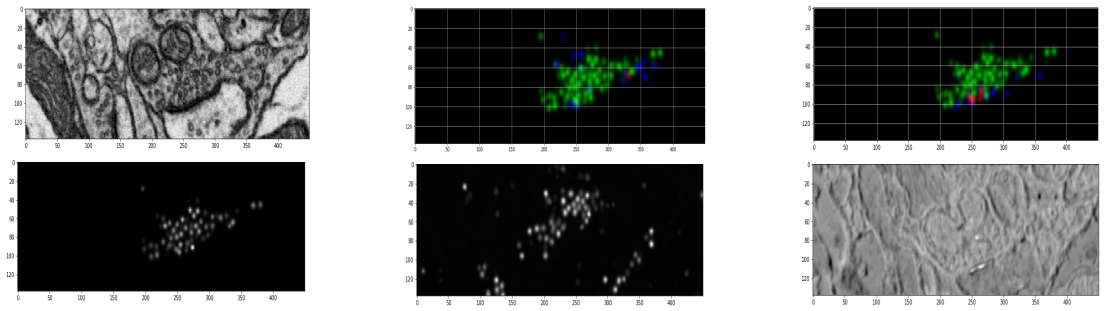
Associated with the previous experiments, we present a set of images where you can see the original images and the results at the output of the network in the different experiments. These results permit us to identify, observe and understand the errors.

In order to understand the figures 4.8 and 4.9, the legend of the colors is:
True positives
False positives
False negative



(a) (b) (c)

Figure 4.8 – **Best stack.** (a) corresponds with the original image and density, (b) the results obtained using two-dimensional images (counting-by-detections and counting-by-density-estimation) and (c) the results obtained using three-dimensional images (counting-by-detections and counting-by-density-estimation).



(a) (b) (c)

Figure 4.9 – **Intermediate stack.** (a) corresponds with the original image and density, (b) the results obtained using two-dimensional images (counting-by-detections and counting-by-density-estimation) and (c) the results obtained using three-dimensional images (counting by detections and counting by density estimation).

Observing the results, using counting-by-detection is better at a qualitative and quantitative level. Working with densities is more complicated and harder to visualise because it produces a lot of noise that difficult its interpretation. The experiment in counting-by-density-estimation probably would be better but the results never will be as good as in counting-by-detection. Given the extraordinary results obtained in the first approach, we decided it would be more significant to improve and invest our time in exploring further the first method.

# 5 Budget

In this section, we quantify the cost to develop this thesis.

You need a computer with technical specification for a data science development and a server with a Nvidia (Titan X).

The necessity to use a server with minimum this specifications is because the images in the data sets are quite large. And for the training is better to use a hight 'hint patch shape' and this requires a lot of memory.

On the other hand, you don't need any license, prototype or specific software. All the code provided in this thesis have an MIT license.

| Concept | Glossary |
|---|---|
| Computer | 1000€ |
| Junior engineer | 12€/hour |
| Number of hour | 400h |
| Total | 5800 € |

Table 5.1 – **Budget.** Costs itemized.

*Itemized of 400 hours = 5h/day*5days/week*4weeks/month*4months

# 6 Conclusion

In this paper, we have proposed two different approaches to count and detect objects in biological images. To this end, we develop two FCN. One works with counting-by-detections and other with counting-by-density-estimation.

We have demonstrated that it is possible to obtain excellent results using counting-by-detection and three-dimensional FCN with crowded and overlapped objects instances. Also, this approach allows computing classic metrics, and it has a better visualisation and interpretation. The easier interpretation was helpful to find problems in the training data and it was fundamental to obtain good results in our experiments.

In this first foray, probably our approach needs a deeper incursion in the part which works with counting-by-density-estimation but the error obtained in counting-by-detection is significantly low, and the value of the precision & recall is much higher than the expected. For this is the reason, we decided to focused on the first approximation.

It would be interesting to analyse the effect of combining the binary cross entropy with density loss term for simultaneous classification and density prediction. This is a possible direction for future work.

# Appendix

## 6.1 Milestones

| WP# | Task# | Short title | Milestone / deliverable | Date(week) |
|---|---|---|---|---|
| 0 | 1 | Classes | Machine learning classes. | 15/01/2018 |
| 0 | 2 | Classes | Homework/project of classes. | 15/01/2018 |
| 1 | 1 | Server connection | Remote connection EPFL. | 21/09/2017 |
| 1 | 2 | Anaconda | Learn how works Anaconda. | 22/09/2017 |
| 1 | 3 | Jupyter | Learn how works Jupyter. | 23/09/2017 |
| 1 | 4 | Tmux | Learn how works Tmux. | 24/09/2017 |
| 2 | 1 | Format of the data | Use data in different formats. | 28/09/2017 |
| 2 | 2 | Vesicles centres | Plot the vesicles centres. | 29/09/2017 |
| 2 | 3 | Dilatation of the centres | Expand the centres. | 06/09/2017 |
| 2 | 4 | Define the uncertainty. | Introduce the information. | 06/09/2017 |
| 2 | 5 | Executable file for the training. | Create an executable | 08/09/2017 |
| 2 | 6 | Results and adjust code. | Visualize and understand. | 11/102017 |
| 3 | 1 | Initial version of the NN. | First intent of the NN. | 16/10/2017 |
| 3 | 2 | Training with 3 vs 1 | Bigger dataset. | 18/10/2017 |
| 3 | 3 | Visualization the results. | Adjust the NN. | 18/10/2017 |
| 3 | 4 | Code: p&rl, f1-score and errors. | Quantify the NN. | 30/10/2017 |
| 3 | 5 | Testing function into the U-net. | Atomization of the testing. | 01/11/2007 |
| 3 | 6 | Understand the errors. | Adjust the NN. | 03/11/2017 |
| 3 | 7 | Training with 7 vs 8. | Training with final dataset. | 05/11/2017 |
| 3 | 8 | Readjust the NN. | Readjust the NN. | 18/11/2017 |
| 3 | 9 | Reduce false positives. | Improve the predictions. | 28/11/2017 |
| 3 | 10 | Final readjust and final training. | Final adjust of the NN. | 30/11/2017 |

Table 3: **Milestones.**

## 6.2 Gantt diagram

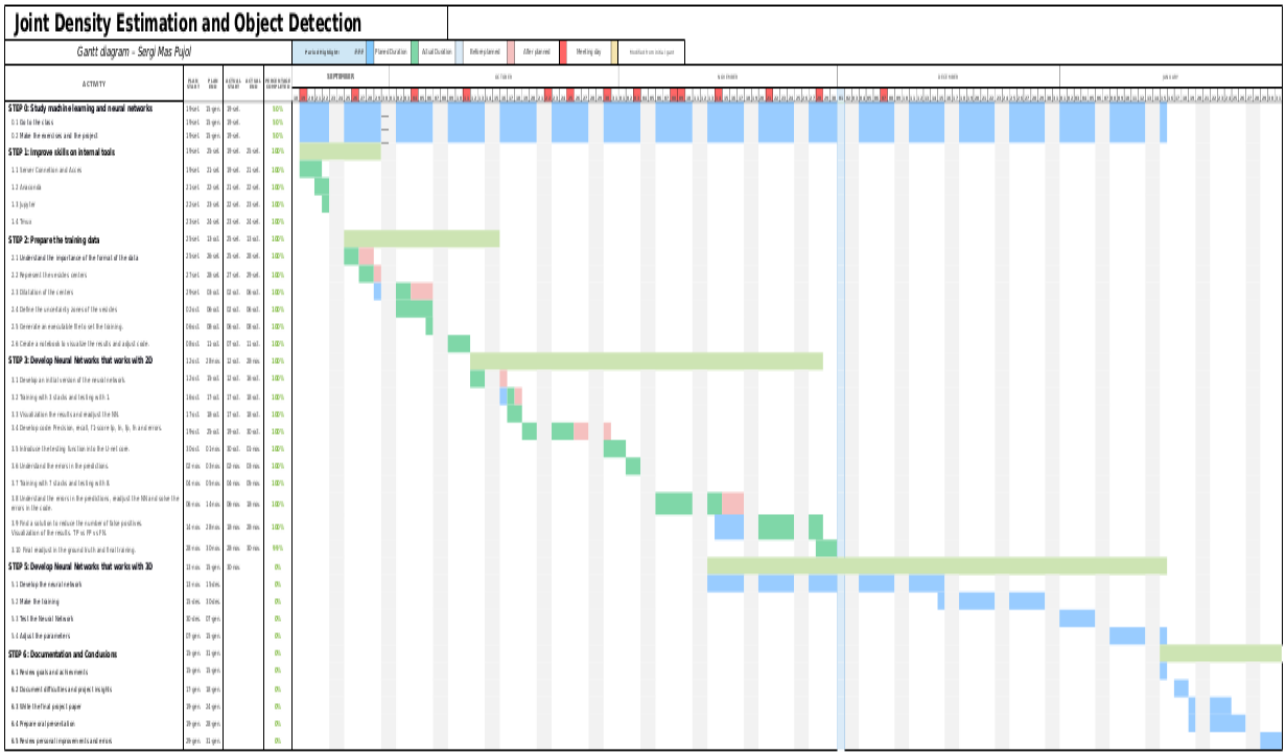



Figure 6.1 – Gantt Diagram

# Bibliography


[1] J. Long, E. Shelhamer, and T. Darrell. Fully convolutional networks for semantic segmentation. 2015.

[2] O. Ronneberger and P.Fischer and T. Brox. U-net: Convolutional networks for biomedical image segmentation. 2015.

[3] V. Lempitsky and A. Zisserman. Learning to count objects in images. 2010.

[4] C. Arteta, V. Lempitsky, J. A. Noble, and A. Zisserman. Interactive object counting. 2014.

[5] Dong C., Loy C.C., He K., Tang X. Learning a deep convolutional network for image super-resolution. 2014.

[6] Boer, PT., Kroese, D.P., Mannor, S. et al. Ann Oper Res. A tutorial on the cross-entropy method. 2005.

[7] Kuhm, H. W. The hungarian method for the assignment problem. 1955.